Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 29 (2012) 3222 – 3226

Procedia Engineering

www.elsevier.com/locate/procedia

2012 International Workshop on Information and Electronics Engineering (IWIEE)

Unilateral View Synthesis Prediction for Even View Rate Overhead Reduce in MVC

Bo Liu[a], Shaohui Liu[a], Yan Liu [a]*, Xiaopeng Fan[a]

[a] *Department of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China*

Abstract

View synthesis is introduced into MVC for improving odd views' coding performance. By synthesizing virtual odd views as reference views, view synthesis prediction (VSP) wins considerable coding gains for odd views. This kind bilateral view synthesis for odd views needs two adjacent views at both sides. Even views have only one reference view according to MVC coding order, and the bilateral view synthesis prediction technique is not suitable then. However, we may try using the only adjacent view to synthesize a virtual view and use it as one additional reference view to improve even views' coding performance. In this paper, we first present a unilateral view synthesis method based on only one adjacent view. To raise the virtual view's quality, we modify our unilateral view synthesis method with Gaussian background modeling. Eventually, we take it into MVC and try to improve even views' coding performance. Experimental results show that our scheme raises the even depth views' coding quality by 0.6 to 1.2 dB, and even texture views less than 0.5dB comparing with MVC.

Keywords: MVC; Holes Filling; View Synthesis; View Synthesis Prediction

1. Introduction

With the development of multiview video (MVV) technology, new applications such as 3D-TV and free-viewpoint (FVV) come up[1]. 3D-TV tries to offer audience a stereoscopic impression, and FVV provides users the ability to interactively select an arbitrary viewpoint. Depth Image Based Rendering (DIBR) algorithm has been recognized as a promising technique to support virtual view rendering. DIBR uses depth map to describe the distance between the object and the camera. By 3D warping using depth

* Corresponding author.
E-mail address: waver@sogou.com, shliu@hit.edu.cn, liuyan@hit.edu.cn.

doi:10.1016/j.proeng.2012.01.470

map, DIBR can synthesize a virtual view at specified position, which is needed by the applications like 3D-TV or FVV.

A multiview plus depth (MVD) format is proposed to represent each view's depth plus texture couple [2]. Multiview Video Coding (MVC) group within MPEG are currently investigating the problems about compression and transmission of such large MVD data. Depth map has many special properties which could be helpful for compression. In ref[3], blocks are described by piecewise-constant functions or piecewise-linear functions, or their combination. Some other works used a transform-based algorithm derived from JPEG-2000 [4] and H.264 encoders [5]. However, MVD data are usually coded with H.264 or MVC directly, in order to be compatible with the existing 2D video processing platform.

MVC outperforms H.264 much by exploiting disparity compensation view prediction (DCVP). Even views take one adjacent view as reference while odd views take two adjacent views as reference. VSP offers a new virtual reference view by bilateral view synthesis. The virtual reference view owns structural similarity with current view, which is intrinsically different from DCVP. Ref [6] shows that performance is considerably improved for odd views by using VSP in MVC.

The rest of this paper is organized as follows. 3D warping and standard bilateral view synthesis are presented firstly in section 2. Our unilateral view synthesis method is proposed in section 3. Experimental results are given in section 4.

2. View Synthesis

To synthesize a virtual view, depth map D[c, t, x, y] is needed to describe the distance from camera c to the object that current pixel location [x, y] represented at time t. Intrinsic matrix A(c) is fixed by the camera itself, while rotation matrix R(c) and translation vector T(c) describe the location of camera c relative to world coordinate [7].

Warping could be separated into two parts. Firstly, apply the well-known pinhole camera model to project location (x, y) in current image coordinates into world coordinates via

$$[u,v,w]=R(c)*A^{-1}(c)*[x,y,1]*D[c,t,x,y]+T(c) \quad (1)$$

Next, an inverse project maps [u, v, w] in world coordinates into the target image coordinates [x', y', z'] via

$$[x',y',z']=A(c')*R^{-1}(c)*\{[u,v,w]-T(c')\} \quad (2)$$

Finally, the coordinates [x', y', z'] are converted to homogenous form [x'/z', y'/z', 1]. The coordinates (x'/z', y'/z') is the target image coordinates. In short, the pixel (x, y) in original image is corresponding to the pixel (x'/z', y'/z') in the virtual view. Warping is just the process of finding out this mapping relationship and view synthesis is to synthesize the virtual view from original image with the mapping relationship.

3. Unilateral View Synthesis

3.1. Unilateral view synthesis flowchart

Unilateral view synthesis is a simplified version of bilateral view synthesis. Taking one path of bilateral view synthesis and added with some postprocess, it forms a unilateral view synthesis flowchart, as fig. 1 shows. Not like bilateral view synthesis, unilateral view synthesis doesn't have two virtual views

for merging, and it causes many holes (the black regions in fig.3 (a) and (d)) need to be filled. We propose an interpolation algorithm to fill the holes and get the virtual view next.

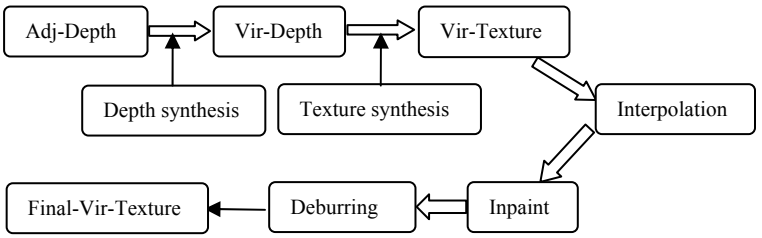


Fig. 1. Unilateral view synthesis flowchart

3.2. The interpolation algorithm

In fig. 2, circles represent the pixels existing and triangles represent the pixels in holes. The black triangle is current pixel that needs to be interpolated and the dotted line is the boundary between hole and non-hole region. Noted that we interpolate hole-pixels in accordance with the sequence left to right and up to down. This interpolation order guarantees that the nearest left pixel of current hole-pixel has already existed and it can be used to interpolate current hole-pixel. The algorithm performs as the following steps. Note that all the depth level comparisons are based on the corresponding depth values.

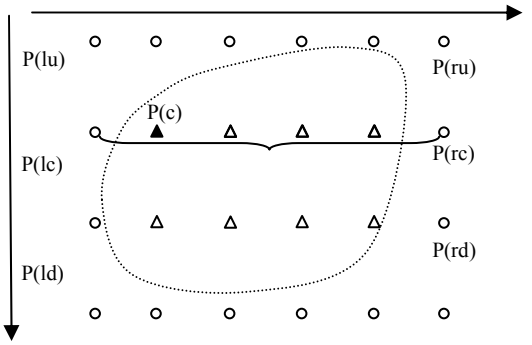


Fig. 2. Interpolation algorithm descriptions

First, if current pixel P(c) is a hole-pixel (not in left or right edge holes), we search for the first right non-hole-pixel P(rc) across the continuous hole-pixels. The nearest left pixel P(lc) is existed according to the interpolation order. Here we set a threshold T to discriminate the foreground and background. In our experiment, we set T as 8. It means that if two pixel values' difference belongs to interval [-8, 8], we think they are in the same depth level. Otherwise, one is foreground and the other is background comparatively.

Second, if P(rc) – P(lc) > T, it means the right region of the hole is foreground and the left region is background. We suppose the hole-pixels between them are all background. Use the left background pixels P(lu), P(lc), P(ld) to interpolate P(c) with the weight (1, 2, 1), which means:

$$P(c)=(P(lu)+P(lc)<<1+P(ld))>>2 \quad (3)$$

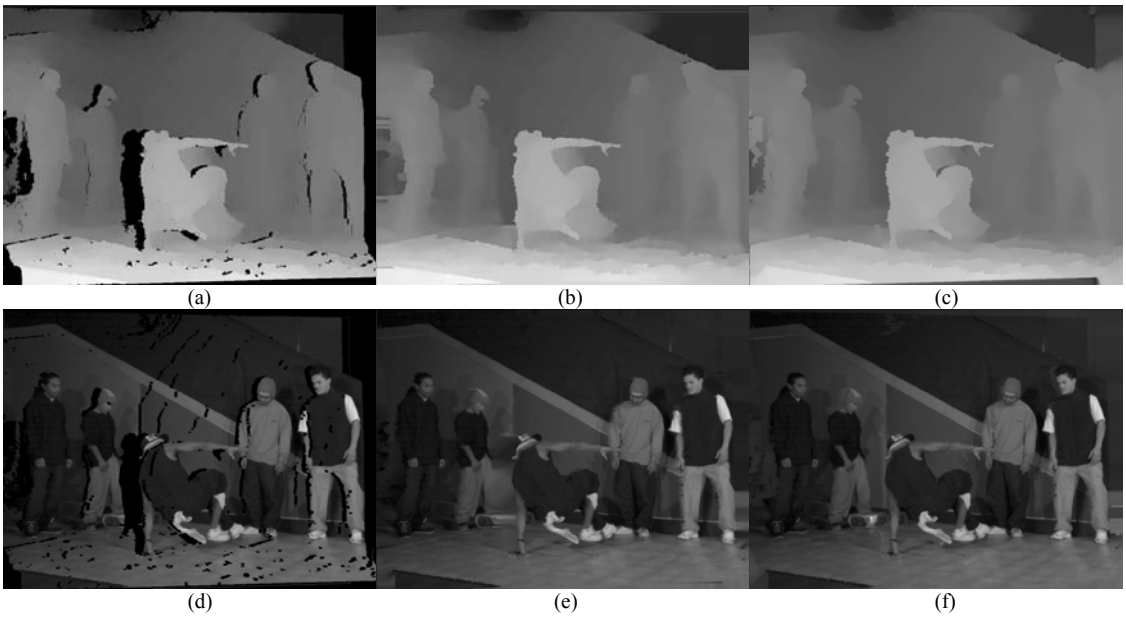


Fig. 3. Subjective of unilateral view synthesis, (a) depth view with holes, (b) depth view using interpolation, (c) depth view using background modeling, (d) texture view with holes, (e) texture view using interpolation, (f) texture view using background modeling

Third, if P(lc) – P(rc) > T, left region is foreground and right is background. Then:

$$P(c)=(P(ru)+P(rc)<<1+P(rd))>>2 \quad (4)$$

Fourth, if abs (P(lc) – P(rc)) <= T, the left region and the right region are in the same depth level. We need judge whether the hole pixels belong to the same depth level as them. Here, we simplify the process and just use linear interpolation to fill the hole pixels. Suppose d1 is the distance between P(c) and P(lc), while d2 is the distance between P(c) and P(rc). Then w1 and w2 are defined by d1 and d2:

$$w_1=d_1/(d_1+d_2),\ w_2=d_2/(d_1+d_2) \quad (5)$$

$$P(c)=w_1*P(lc)+ w_2*P(rc) \quad (6)$$

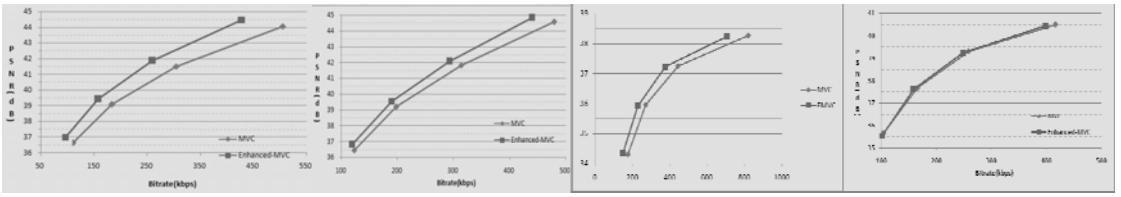
Fig. 3 (b) and (e) show the virtual depth view2 and virtual texture view2 using this method.

3.3. Modify with Gaussian background modeling

To improve virtual view's quality, we use Gaussian background modeling [8] to obtain a background image of current view. The background image offers more accurate details of the holes in virtual texture view. It recovers the hole details much better than interpolation directly. Fig. 3 (c) and (f) show that the details in some holes are recovered well and the subjective quality is much better than the case obtained by interpolation directly.

3.4. MVC using unilateral VSP for even views

By taking unilateral VSP into MVC, we can get one additional virtual reference view for even views. It means each even view, e.g. view2, owns both one DCVP reference view and one VSP reference view. This makes MVC encode even views of MVD data better, as fig.4 shows.



(a) (b) (c) (d)

Fig. 4. Using unilateral VSP in MVC, view2 (a) breakdancers, depth (b) ballet, depth (c) breakdacners, texture (d) ballet, texture

4. Experimental Results

Experiments are conducted using the first 25 frames of view0 of breakdancer and ballet sequence at 15Hz. All the views are encoded according to the MVC common conditions [9] except that GOP size is 12. We use the coded view0 to synthesize a virtual view of view2 and take the virtual view2 as reference view when coding view2. Our experiments are based on the JMVM 2.1 software.

Fig. 4 shows the results of encoding view2 with QPs 26, 30, 34 and 38. The blue lines are the results of original MVC, and red lines are the results using the virtual view2 as additional reference views. Fig.4 (a) and (b) show that unilateral VSP improves the even depth views' coding performance greatly comparing with MVC. The gain is more than 1db for breakdancer and more than 0.5db for ballet. Texture views can also improve a little, as (c) and (d) shows, which depends on the sequence's properties.

This work is supported by Major State Basic Research Development Program of China (973 Program) (2009CB320905), the Natural Science Foundation of China (60803147, 60736043) and the Fundamental Research Funds for the Central Universities (HIT.NSRIF.2009068).

References

[1] Introduction to 3D Video. *ISO/IEC/JTC1/SC29/WG11*. Doc.N9784. Archamps, France; May 2008.

[2] P. Merkle, A. Smolic, K. Mller, T. Wiegand. Multi-view Video plus Depth Representation and Coding. In: *Image Proc. of ICIP*, San Antonio, TX, Dec. 2007, p. 201-204

[3] R. Krishnamurthy, B-B. Chai, H. Tao, and S. Sethuraman. Compression and transmission of depth maps for image-based rendering. In: *IEEE Int. Conf. on Image Proc*, vol. 3, Thessaloniki, Greece, Oct. 2001, p. 828–831

[4] Y. Morvan, D. Farin, P. H.N. de With. Depth-Image Compression based on an R-D Optimized Quadtree Decomposition for the Transmission of Multiview Images. In: *J Sci Image Proc. of ICIP*, San Antonio, TX, Dec. 2007, p. 201-204

[5] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Mueller, P. H. N. de With, and T. Wiegand. The Effect of Depth Compression on Multiview Rendering Quality. In: *IEEE Conference on 3D-TV*, Istanbul, Turkey, May 2008, p. 245-248

[6] E. Martinian, A. Behrens, J. Xin, A. Vetro and H. Sun. Extensions of H.264/AVC for Multiview Video Compression. In: *IEEE Int'l Conf. Image Proc.*, Atlanta, GA, Oct. 2006.

[7] Mateusz Gotfryd, Krzysztof Wegner and Marek Domański. View synthesis software and assessment of its performance. *ISO/IEC JTC1/SC29/WG11, MPEG/M15672*, Hannover, Germany, July 2008.

[8] P. KaewTraKulPong, R. Bowden. n Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. In: *Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, AVBS01, Sept 2001.

[9] Y. Su, A. Vetro and A. Smolic. Common Test Conditions for Multiview Video Coding. *JVT-T207*, Klagenfurt, Austria, July 2006.